SPECIFICATION

Electronic Version 1.2.8

Stylesheet Version 1.0

METHOD AND SYSTEM FOR DOCUMENT IMAGE LAYOUT DECONSTRUCTION AND REDISPLAY SYSTEM

Background of the Invention

[Field of Invention]

[0001] The invention relates generally to the problem of making an arbitrary document, conveniently readable on an arbitrarily sized display.

[Description of Related Art]

[0002] Existing systems for rendering page-image versions of documents on display screens have required manual activities to improve the rendering, or clumsy panning mechanisms to view direct display of page images on wrong-sized surfaces. In particular, it has been necessary to either (1) key in the entire text manually, or (2) process the page images through an optical character recognition (OCR) system and then manually tag the resulting text in order to preserve visually important layout features.

[0003] Problems with existing systems include: (a) high expense of manual keying and/or correcting of OCR results and manual tagging; (b) the risk of highly visible and disturbing errors in the text resulting from OCR mistakes; and (c) the loss of meaningful or aesthetically pleasing typeface and type size choices, graphics and other non-text elements; and (d) loss of proper placement of elements on the page.

[0004] Such problems are significant, for example, because book publishers are increasingly creating page-image versions of books currently being published, as well

as books from their backlists. The page-image versions are being created for print-on-demand usage. While print-on-demand images can be re-targeted to slightly larger or slightly smaller formats by scaling the images, they cannot currently be re-used for most electronic book purposes without either re-keying the book into XML format, or scanning the page images using OCR and manually correcting the re-keyed and scanned images.

Summary of Invention

[0005] The invention provides methods and systems for converting any document originating in a page-image format, such as a scanned hardcopy document represented as a bitmap, into a form suitable for display on screens of arbitrary size, through automatic reformatting or "reflowing" of document contents.

[0006] Reflowing is a process that moves text elements (often words) from one text-line to another so that each line of text can be contained within given margins. Reflowing typically breaks or fills lines of text with words, and may re-justify column margins, so that the full width of a display is used and no manual 'panning' across the text is needed. As an example, as a display area, within which lines of text appear, is altered so that the width of the visible text is reduced, it may be necessary for words to be moved from one text-line to another to shorten the length of all of the text-lines so that no text-line is too long to be entirely visible in the display area. Conversely, if the display area is widened, words may be moved from one text-line to another so that the length of text-lines increase, thereby allowing more text-lines to be seen without any word image being obscured.

[0007] Image and layout analysis transforms the raw document image into a form that is reflowable and that can be more compactly represented on hand-held devices. In various exemplary embodiments, image analysis begins with adaptive thresholding and binarization. For each pixel, the maximum and minimum values within a region around that pixel, are determined using greyscale morphology. If the difference between these two values is smaller than a statistically determined threshold, the region is judged to contain only white pixels. If the difference is above the threshold, the region contains both black and white pixels, and the minimum and maximum values represent the blank ink and white paper background values, respectively. In the

first case, the pixel value is normalized by bringing the estimated white level to the actual white level of the display. In the second case, the pixel value is normalized by expanding the range between the estimated white and black levels to the full range between the white level and the black level of the display. After this normalization process, a standard thresholding method can be applied.

[0008] In the thresholded image, connected components are labeled using a scan algorithm combined with an efficient union-find data structure. Then, a bounding box is determined for each connected component. This results in a collection of usually several thousand connected components per page. Each connected component may represent a single character, a portion of a character, a collection of touching characters, background noise, or parts of a line drawing or image. These bounding boxes for connected components are the basis of the subsequent layout analysis.

[0009] In various exemplary embodiments, for layout analysis, the bounding boxes corresponding to characters in the running text of the document, as well as in a few other page elements, such as, for example, headers, footers, and/or section headings, are used to provide important information about the layout of the page needed for reflowing. In particular, the bounding boxes and their spatial arrangement identify page rotation and skew, column boundaries, what tokens may be needed for token-based compression, reading order, and/or how the text should flow between different parts of the layout. Bounding boxes that are not found to represent "text" in this filtering operation are not lost, however. Such bounding boxes can later be incorporated into the output from the system as graphical elements.

[0010] The dimensions of bounding boxes representing body text are found using a simple statistical procedure. Using the distribution of heights as a statistical mixture of various components, for most pages containing text, the largest mixture component often corresponds to lower case letters at the predominant font size. The size is used to find the x-height of the predominant font and the dimension is used to filter out bounding boxes that are either too small or too large to represent body text or standard headings.

[0011] Given a collection of bounding boxes representing text, it is desirable to find text lines and column boundaries. The approach used in various exemplary embodiments

to identify text lines and column boundaries relies on a branch-and-bound algorithm that finds maximum likelihood matches against line models under a robust least square error model, i.e., a Gaussian noise model in the presence of spurious background features. Text line models are described by three parameters: the angle and the offset of the line, and the descender height. Bounding boxes whose alignment point, that is, the center of the bottom side of the bounding box, rests either on the line or at a distance given by the descender height below the line, are considered to match the line. Matches are penalized by the square of their distance from the model, up to a threshold value ϵ , which is usually on the order of five pixels.

[0012] After a text line has been found, the bounding box that bounds all of the connected components that participated in the match is determined. All other connected components that fall within that bounding box are assigned to the same text line. This tends to "sweep up" punctuation marks, accents, and "i"-dots that would otherwise be missed. Within each text line, multiple bounding boxes whose projections onto the baseline overlap are merged. This results in bounding boxes that predominantly contain only or more complete characters, as opposed to bounding boxes that contain only or predominantly portions of characters. The resulting bounding boxes are then ordered by the x-coordinate of the lower left corner of the bounding boxes to obtain a sequence of character images in reading order. Multiple text lines are found using a greedy strategy, in which the top match is first identified. Then, the bounding boxes that participated in the match are removed from further consideration. Next, the next best text line is found, until no good text line matches can be identified anymore.

[0013] This approach to text line modeling has several advantages over known projection or linking methods. First, different text lines can have different orientations. Second, by taking into account both the baseline and the descender line, the technique can find text lines that are missed by known text line finders. Third, the matches returned by this method follow the individual text lines more accurately than other known methods.

[0014]

Column boundaries are identified in a similar manner by finding globally optimal maximum likelihood matches of the center of the left side of bounding boxes against

a line model. In order to reduce background noise, prior to applying the line finder to column finding, statistics about the distribution of horizontal distances between bounding boxes are used to estimate the intercharacter and inter-words spacing, i.e., the two largest components in the statistical distribution of horizontal bounding box distances. The bounding boxes for characters are then merged into words. This reduces severalfold the number of bounding boxes that need to be considered for column matching and tends to improve the reliability of column boundary detection.

[0015] Any connected components that are not part of a text line are grouped together and treated as images. For a single column document, by enumerating text lines and bounding boxes of images in order of their y-coordinates, a sequence of characters, whitespaces, and images in reading order is obtained. For a double column document, the two columns are treated as if the right column were placed under the left column.

[0016] This simple layout analysis technique copes with a large number of commonly occurring layouts in printed documents and transform such layouts into a sequence of images that can be reflowed and displayed on a smaller-area display device. The simple technique works well in these applications because the requirements of reflowing for a smaller-area display device, such as a document reader, are less stringent than for other layout analysis tasks, like rendering into a word processor. Since the output of the layout analysis will only be used for reflowing and not for editing, no semantic labels need to be attached to text blocks. Because the documents are reflowed on a smaller area screen, there is also no user expectation that a rendering of the output of the layout analysis precisely match the layout of the input document. Furthermore, if page elements, like headers, footers, and/or page numbers, are incorporated into the output of the layout analysis, users can easily skip such page elements during reading. Such page elements may also serve as convenient navigational signposts on the smaller-area display device.

[0017]

In various exemplary embodiments, the methods and systems according to this invention more specifically provide a two-stage system which analyzes, or "deconstructs", page image layouts. Such deconstruction includes both physical, e.g., geometric, and logical, e.g., functional, segmentation of page images. The segmented image elements may include blocks, lines, and/or words of text, and other segmented

image elements. The segmented image elements are then synthesized and converted into an intermediate data structure, including images of words in correct reading order and links to non-textual image elements. The intermediate data structure may, for example, be expressed in a variety of formats such as, for example, Open E-book XML, Adobe ™ PDF 1.4 or later, HTML and/or XHTML, as well as other useful formats that are now available or may be developed in the future. In various exemplary embodiments, the methods and systems according to this invention then distill or convert, the intermediate data structure for "redisplay" into any of a number of standard electronic book formats, Internet browsable formats, and/or print formats.

[0018] In various exemplary embodiments of the methods and systems according to this invention, the intermediate data structure may contain tags, such as those used in SGML and XML, which state the logical functions or geometric properties of the particular image elements the tags annotate. It is also possible that, in various exemplary embodiments, some image elements may not have tags attached to them. For example, in instances where the functions and properties of image elements may be inferable from their position and the position of other tagged and untagged image elements in the intermediate data structure, such tags may not be necessary.

[0019] It is also possible that, in various exemplary embodiments, special image elements that can be used for this purpose are not extracted from the original page image, but are created as tagged or untagged elements. Such special image elements can be inserted into the intermediate data structure in an order that would define the desired functions and properties of other image elements. For example, a special image element may be a blank that represents a space between two words. Further, special non-image markers, other than tags attached to particular image elements, could be inserted so that the functions and properties of at least some of the image elements may be inferred from their relative position with respect to the markers within the intermediate data structure.

[0020] To prepare the intermediate data structure for redisplay, the intermediate data structure may be converted, for example, to HTML for use on a standard Internet browser, or to Open E-book XML format for use on an Open E-book reader. Other methods may include, for example, converting the intermediate data structure to

Plucker format for use on a Plucker electronic book viewer, or to Microsoft Reader format for display using MS Reader format or to a print format for printing to paper or the like.

[0021] In any document image, the physical layout geometry is fixed and the logical or functional layout structure is implicit. That is, it is intended to be understood by human readers, who bring, to the task of reading, certain conventional expectations of the meaning and implications of layout, typeface, and type size choices. In various exemplary embodiments, in the intermediate data structure according to the methods and systems of this invention, by contrast, the original fixed positions of words are noted but not strictly adhered to, so that the physical layout becomes fluid. In various exemplary embodiments, aspects of the logical structure of the document are captured explicitly, and automatically, and represented by additional information. In various exemplary embodiments, the intermediate data structure according to this invention is automatically adaptable at the time of display to the constraints of size, resolution, contrast, color, geometry, and/or the like, of any given display device or circumstance of viewing.

[0022] The adaptability enabled by the methods and systems according to this invention include re-pagination of text, reflowing, such as, for example, re-justification, reformatting, and/or the like, of text into text-lines, and logical linking of text to associated text and/or non-text contents, such as illustrations, figures, footnotes, signatures, and/or the like. In various exemplary embodiments, the methods and systems according to this invention take into account typographical conventions used to indicate the logical elements of a document, such as titles, author lists, body text, paragraphs, and/or hyphenation, for example. In various exemplary embodiments, the methods and systems of the invention also allow the reading order to be inferred within blocks of text and/or among blocks of text on the page.

[0023] Thus, redisplaying the document is enabled for a wide range of displays whose size, resolution, contrast, available colors, and/or geometries may require the document's contents to be reformatted, reflowed, re-colored, and/or reorganized to achieve a high degree of legibility and a complete understanding of the document's contents, without requiring OCR or re-keying, and without being subject to the

respective attendant errors of OCR or re-keying, and without losing the look and feel of the original document as chosen by the author and publisher.

[0024] In various exemplary embodiments, the methods and systems according to this invention reduce costs by obviating the need for manual keying, correction of OCR results, and/or tagging. In various exemplary embodiments, the methods and systems according to this invention tend to avoid introducing OCR character recognition errors. In various exemplary embodiments, the methods and systems according to this invention tend to preserve typeface and type size choices made by the original author and publisher, which may be helpful, or even essential, in assisting the reader in understanding the author's intent. In various exemplary embodiments, the methods and systems according to this invention also tend to preserve the association of graphics and non-textual elements with related text.

[0025] These and other features and advantages of this invention are described in, or are apparent from, the following detailed description of various exemplary embodiments of the systems and methods according to this invention.

Brief Description of the Drawings

[0026] Various exemplary embodiments of the systems and methods according to this invention will be described in detail, with reference to the following figures, wherein:

[0027] Figure 1 illustrates an intermediate representation of an image of a page, using XHTML;

[0028] Figure 2 illustrates the format and content of the intermediate representation without the use of tags or explicit separators;

[0029] Figure 3 is a flowchart outlining one exemplary embodiment of a method for document image layout deconstruction and redisplay;

[0030] Figure 4 is a block diagram of one exemplary embodiment of a document deconstruction and display system according to this invention.

Detailed Description of Exemplary Embodiments

[0031] Figure 1 illustrates a detailed example of an intermediate data structure 260 for a

page image 300. In Fig. 1 the intermediate data structure 260 is expressed using XHTML as an example of an intermediate data structure format. The page image 300 is shown schematically having a first text area 310 which functions as a title, a second area 320 which functions as an author list, third text areas 330 which function as paragraphs, and a fourth text area 340 which functions as a page number. The structures represented by these text areas 310-340 are usually significant to both the author and the reader, and so are detected and preserved in the intermediate data structure 260. For example, the intermediate data structure 260 preserves the title text area 310 by noting the position of this title text area 310 at the top of the page image, that the text area 310 is centered, and the large typeface used in this text area 310. The position is preserved in the intermediate data structure 260 by the XHTML tag "<DIV CLASS=title ID=title>". Also, the intermediate data structure 260 preserves the author-list text area 320 by the position, of this author-list text area 320 just beneath the title text area 310. The intermediate data structure 260 preserves the centered position of the author-list text area 320, and that the author-list text area 320 is printed in a large typeface that is smaller than the typeface of the title text area 310. In particular, in the specific exemplary embodiment shown in Fig. 3, the author-list text area 320 is preserved in the intermediate data structure 260 by the XHTML tag "<DIV CLASS=authors ID=authors>".

[0032] Figure 2 shows a representation of the page image 300 as a sequence of image elements 190, and the corresponding representative compressed image tokens 200, without using attached tags or explicit separators. For example, in a document where the functions and properties of image elements may be inferable from their position on the page and the position of other tagged and untagged image elements in the intermediate data structure, it is not necessary to tag all of the image elements.

[0033] Figure 3 is a flowchart outlining one exemplary embodiment of a method for document image layout deconstruction and redisplay. As shown in Fig. 3, operation of the method begins in step S100 and continues to step S110, where a document is input by scanning, or use of another data source that provides a document that is in a page image format. The document may be represented as a set of page images, such as bi-level, gray-scale, or as color images, in one of a set of image file formats such as TIFF and JPEG, for example.

[0034] Then, in step S120, the image file of the page image is analyzed to identify text image areas and non-text image areas. Text area images may include, for example, blocks (or columns), lines, words, or characters of text. Non-text area images may include, for example, illustrations, figures, graphics, line-art, photographs, handwriting, footnotes, signatures and/or the like.

[0035] Next, in step S130, the identified text image areas and non-text image areas are located and isolated. Locating and isolating text image areas may include, for example, locating and isolating the baseline and, possibly, top-line and/or cap-line, of each text line image. The isolated line regions are modeled as line segments that run from one end of the text line image to another. Baselines may be modeled as straight lines which are horizontal or, in the case of Japanese, Chinese, and other scripts, vertical, or oriented at some angle near the horizontal or the vertical. Baselines may also be modeled as curved functions. Operation then continues in step S140.

[0036] In step S140, the isolated text image areas are selected for further processing. Next, in step S150, the text line regions of the selected text image areas are located and isolated and the layout properties of the selected text image areas are then determined. Layout properties may include, for example, indentation, left and/or right justification, centering, hyphenation, special spacing (e.g. for tabular data), proximity to figures and other non-textual areas, and the like. Layout properties may also include type size and typeface-family properties (e.g. roman/bold/italic styles) that may indicate the function of the text within the page. Operation then continues in step S160.

[0037] In step S160, the located text line regions are further processed into a set of segmented image elements. Then in step S170, the segmented image elements are read and basic textual elements are located and isolated. Basic textual elements may include, for example, words, numbers, dates, proper names, bibliographic references, references to figures, and/or other non-textual elements within or outside the document. The textual elements will become the basic image units which will be reflowed and reconstructed in later stages. As part of locating the segmented image elements, each segmented image element is labeled with the position of the element

relative to the baseline of the text line so that when the text-lines are later reflowed, the reconstructed baseline may be referred to when placing the corresponding segmented image elements so the elements appear to share the newly constructed baseline. Operation then continues to step S180.

[0038] In step S180, the set of segmented image elements are labeled with their baseline-relative position. Next, in step S190, the segmented image elements and the relative baselines portions are compressed into token-based image elements. Then, in step S200, the image elements are synthesized into an intermediate data structure. Operation then continues to step S210.

[0039] In step S210, the intermediate data structure is stored to retain the data in an intermediate format until distilling and redisplay is desired. Then, in step S220, the stored data is distilled to convert the data into a device specific display format. The intermediate data structure may be converted, for example, to HTML for use on a standard Internet browser, or to Open E-book XML format for use on an Open E-book reader. Other methods may include, for example, converting the intermediate data structure to Plucker format for use on a Plucker electronic book viewer, or to Microsoft Reader format for display using MS Reader format or to a print format for printing to paper or the like. Next, in step S230, the distilled data is displayed to the user. Operation of the method then continues to step S240, where operation of the method ends.

[0040] In various exemplary embodiments of this invention, the intermediate data structure may also be in a form that can be processed by an E-Book distiller for redisplaying the intermediate data structure on an E-book reader. In the event the intended use is to display an electronic book, then an E-book distiller reads the intermediate data structure and prepares it for display on a specific device such as a PDA, a computer graphical interface window, or any other graphical display device. Such processing of the intermediate data structure is not limited to an E-Book distiller, but may accomplished be any method or device for re-converting the intermediate data structure for redisplay on a selected display device.

[0041] In various exemplary embodiments of this invention, the intermediate data structure may be expressed in a variety of formats such as, for example, Open E-book

XML, Adobe ™ PDF 1.4 or later, HTML and/or XHTML, as well as other useful formats that are now available or may be developed in the future. In various exemplary embodiments of this invention, the intermediate data structure may contain tags, such as those used in SGML and XML.

[0042] In various exemplary embodiments, in step S190, the segmented image elements are compressed into a smaller number of prototype images, so that each incoming element may be replaced by a prototype that is visually similar to, or perhaps indistinguishable from the image elements. This is an instance of 'token-based' compression where the tokens are the image elements. Therefore, if the image elements are words, then the tokens are words. Alternatively, it may be advantageous to cut the image elements into smaller images corresponding exactly or approximately with individual characters since there are fewer distinct characters than words in some languages. Compressing the segmented image elements may further include writing a set, or dictionary, of representative compressed image tokens, and a list of references into the representative compressed image tokens. Each reference represents an original image element labeled with its position relative to the baseline.

[0043] In various exemplary embodiments of this invention, the non-text image areas, compressed non-text image areas, the set of representative compressed image tokens, the segmented image elements and/or the layout characteristics are synthesized in step S200 into an intermediate data structure. However, in various exemplary embodiments of this invention, non-text area images may optionally first be compressed in step S190, for file compression, before being synthesized in step S200 for integration into the intermediate data structure. Additionally, in various exemplary embodiments of this invention, the segmented image elements may be optionally compressed in step S190 before being synthesized in step S200 for integration into the intermediate data structure. Determining whether to compress the non-text image areas and the segmented image elements may be dependent on file size or other user specific parameters. If the intermediate data structure does not include compressed data, then the intermediate data structure may be represented as XHTML, for example.

[0044] In various exemplary embodiments of this invention, the intermediate data

structure may also contain a tagged list containing references to every textual and non-image element that are proximate to or references by textual image element as well as layout characteristics such as indentation, hyphenation, spacing, and the like. In addition to this list, a set of representative compressed image tokens can be written to a separate but intimately associated image element database. The intermediate data structure contains all the information required to support the reflowing and the reconstruction of the image elements.

[0045] Figure 4 is a block diagram of one exemplary embodiment of a document deconstruction and redisplay system 400 according to this invention. As shown in Fig. 4, one or more user input devices 480 are connected over one or more links 482 to an input/output interface 410. Additionally, a data source 500 is connected over a link 502 to the input/output interface 410. A data sink 600 is also connected to the input/output interface 410 through a link 602.

[0046] Each of the links 482, 502, 602 can be implemented using any known or later developed device or system for connecting the one or more user input devices 480, the data source 500 and the data sink 600, respectively, to the document layout deconstruction and redisplay system 400, including a direct cable connection, a connection over a wide area network or a local area network, a connection over an intranet, a connection over the Internet, or a connection over any other distributed processing network or system. In general, each of the links 482, 502, 602 can be any known or later developed connection system or structure usable to connect the one or more user input devices 480, the data source 500 and the data sink 600, respectively, to the document layout deconstruction and redisplay system 400.

[0047] The input/output interface 410 inputs data from the data source 500 and/or the one or more user input devices 480 and outputs data to the data sink 600 via the link 602. The input/output interface 410 also provides the received data to one or more of the controller 420, the memory 430, a deconstructing circuit, routine or application 440, a synthesizing circuit, routine or application 450, a distilling circuit, routine or application 460, and/or a display 490. The input/output interface 410 receives data from one or more of the controller 420, the memory 430, the deconstructing circuit, routine or application 440, the synthesizing circuit, routine or application 450, and/or

the distilling circuit, routine or application 460.

[0048] The memory 430 stores data received from the deconstructing circuit, routine or application 440, synthesizing circuit, routine or application 450, the distilling circuit, routine or application 460, and/or the input/output interface 410. For example, the original data, the deconstructed data, the synthesized data, and/or the distilled data, may be stored in the memory 430. The memory can also store one or more control routines used by the controller 420 to operate the document layout deconstruction and redisplay system 400.

[0049] The memory 430 can be implemented using any appropriate combination of alterable, volatile or non-volatile memory or non-alterable, or fixed, memory. The alterable memory, whether volatile or non-volatile, can be implemented using any one or more of static or dynamic RAM, a floppy disk and disk drive, a writable or re-writeable optical disk and disk drive, a hard drive, flash memory or the like. Similarly, the non-alterable or fixed memory can be implemented using any one or more of ROM, PROM, EPROM, EEPROM, an optical ROM disk, such as a CD-ROM or DVD-ROM disk, and disk drive or the like.

[0050] It should be understood that each of the circuits or routines shown in Fig. 4 can be implemented as portions of a suitably programmed general purpose computer. Alternatively, each of the circuits or routines shown in Fig. 4 can be implemented as physically distinct hardware circuits within an ASIC, or using a FPGA, a PDL, a PLA or a PAL, or using discrete logic elements or discrete circuit elements. The particular form each of the circuits or routines shown in Fig. 4 will take is a design choice and will be obvious and predicable to those skilled in the art.

[0051] In operation, the data source 500 outputs a set of original data, i.e., input document, scanned document, or the like, over the link 502 to the input/output interface 410. Similarly, the user input device 480 can be used to input one or more of a set of newly created original data, scanned data, or the like, over the link 482 to the input/output interface 410. The input/output interface 410 directs the received set of data to the memory 430 under the control of the controller 420. However, it should be appreciated that either or both of these sets of data could have been previously input into the document layout deconstruction and redisplay system 400.

[0052] An input document is input into the deconstructing circuit, routine or application 440 under control of the controller 420. The deconstructing circuit, routine or application 440 reads image files and locates and isolates text area images and non-text area images. Non-text area images are then sent to the synthesizing circuit, routine or application 450 under control of the controller 420 for synthesizing the data into an intermediate data structure. Non-text images may optionally be compressed prior to being synthesized at the synthesizing circuit, routine or application 450.

[0053] The deconstructing circuit, routine or application 440 reads the set of isolated images text area images and locates and isolates text line regions and detects the layout properties of the text line regions. The layout properties are sent to the synthesizing circuit, routine or application 450 under the control of the controller 420. The text line regions are further processed by the deconstructing circuit, routine or application 440 into a set of segmented image elements with their baseline relative portions and then sent to the synthesizing circuit or routine 450 under control of the controller 420 for synthesizing into an intermediate data structure. The deconstructing circuit, routine or application 440 may also compress the segmented image elements with their baseline relative portions into token-based image elements before being sent to the synthesizing circuit, routine or application 450 under control of the controller 420 for synthesizing into an intermediate data structure.

[0054] It should be appreciated that the deconstructing circuit, routine or application 440 and the synthesizing circuit, routine or application 450 can use any known or later-developed encoding scheme, to deconstruct and synthesize the data to be converted into an intermediate data structure that may then be distilled by the distilling circuit, routine or application 460 for display on the display device 490.

[0055] The synthesizing circuit, routine or application 450 synthesizes the non-text area images and compressed non-text area image elements, the set of representative compressed image tokens, the segmented image elements and the layout characteristics, and transcribes the data into an intermediate data structure. The intermediate data structure is sent to the memory 430 under the control of the controller 430 for storage.

[0056] Upon request by a user of the input document, the distilling circuit, routine or application 460 converts the intermediate data structure into a format usable by the display 490. The distilling circuit, routine or application 460, under control of the controller 420 and the input output interface 410, will output the converted intermediate data structure to the user's device for display.

[0057] It should be appreciated that the distilling circuit, routine or display 460 can use any known or later-developed encoding scheme, including, but not limited to, those disclosed in this application, to convert the intermediate data structure into a device specific format usable for redisplay on an arbitrarily sized display.

[0058] In various exemplary embodiments, the systems and methods of this invention also relate to the use of special non-image markers, other than tags attached to particular image elements, to infer the functions and properties of all the image elements from their relative positions with respect to the markers within the intermediate data structure.

[0059] While this invention has been described in conjunction with the exemplary embodiments outlined above, it is evident that many alternatives, modifications and variations will be apparent to those skilled in the art. Accordingly, the exemplary embodiments of the invention, as set forth above, are intended to be illustrative, not limiting. Various changes may be made to the invention without departing from the spirit and scope thereof.